Conditional Bisimilarity for Reactive Systems

Mathias Hülsbusch
Universität Duisburg-Essen, Germany

Barbara König
Universität Duisburg-Essen, Germany
barbara_koenig@uni-due.de

Sebastian Küpper
FernUniversität in Hagen, Germany
sebastian.kuepper@fernuni-hagen.de

Lars Stoltenow
Universität Duisburg-Essen, Germany
lars.stoltenow@uni-due.de

Abstract

Reactive systems à la Leifer and Milner, an abstract categorical framework for rewriting, provide a suitable framework for deriving bisimulation congruences. This is done by synthesizing interactions with the environment in order to obtain a compositional semantics.

We enrich the notion of reactive systems by conditions on two levels: first, as in earlier work, we consider rules enriched with application conditions and second, we investigate the notion of conditional bisimilarity. Conditional bisimilarity allows us to say that two system states are bisimilar provided that the environment satisfies a given condition. We present several equivalent definitions of conditional bisimilarity, including one that is useful for concrete proofs and that employs an up-to-context technique, and we compare with related behavioural equivalences. We instantiate reactive systems in order to obtain DPO graph rewriting and consider a case study in this setting.

2012 ACM Subject Classification Theory of computation → Concurrency; Theory of computation → Program reasoning

Keywords and phrases conditional bisimilarity, reactive systems, up-to context, graph transformation

Digital Object Identifier 10.4230/LIPIcs.FSCD.2020.10

Related Version A full version of the paper [17] is available at `https://arxiv.org/abs/2004.11792`.

Funding Research partially supported by DFG Project BEMEGA.

1 Introduction

Behavioural equivalences, such as bisimilarity, relate system states with the same behaviour. Here, we are in particular interested in conditional bisimilarity, which allows us to say that two states a, b are bisimilar provided that the environment satisfies a condition $\mathcal{C}$. Work on such conditional bisimulations appears somewhat scattered in the literature (see for instance [21, 15, 11, 3]). They also play a role in the setting of featured transition systems for modelling software product lines [7], where the behaviour of many products is specified in a single transition system. In this setting it is possible to state that two states are bisimilar for certain products, but not for others.

We believe that conditional notions of behavioural equivalence are worthy of further study. In practice it may easily happen that two sub-systems are only ever used in restricted environments and it is too much to ask that they behave equivalently under all possible contexts. Furthermore, instead of giving a simple yes/no-answer, bisimulation checks can answer in a more fine-grained way, specifying conditions which ensure bisimilarity.

We state our results in a very general setting: reactive systems à la Leifer and Milner [22], a categorical abstract framework for rewriting, which provides a suitable framework for deriving bisimulation congruences. In particular, this framework allows to synthesize labelled transitions from plain reaction rules, such that the resulting bisimilarity is automatically a congruence. Intuitively, the label is the minimal context that has to be borrowed from the environment in order to trigger a reduction. (Transitions labelled with such a minimal context will be called representative steps in the sequel. They are related to the idem pushout steps of [22].) Here, we rely on the notion of saturated bisimilarity introduced in [5] and we consider reactive system rules with application conditions, generalizing [16].

Important instances of reactive systems are process calculi with contextualization, bigraphs [18] and double-pushout graph rewriting [8], or in general rewriting in adhesive categories [20]. Hence we can use our results to reason about process calculi as well as dynamically evolving graphs and networks for various different types of graphs (node- or edge-labelled graphs, hypergraphs, etc.). Our contributions in this paper can be summarized as follows:

- We define the notion of conditional bisimilarity, in fact we provide three equivalent definitions: two notions are derived from saturated bisimilarity, where a context step (or a representative step) can be mimicked by several answering steps. Third, we compare with the notion of conditional environment congruence, which is based on the idea of annotating transitions with passive environments enabling a step.
- Conditional bisimulation relations tend to be very large – often infinite in size. In order to handle conditional bisimulation, we propose an up-to context technique that allows to replace infinite conditional bisimulations by possibly finite bisimulations up-to context, which provide witnesses for bisimilarity.
- We compare conditional bisimilarity with related notions of behavioural equivalence.
- To illustrate our concepts, we work out a small case study in the context of double-pushout graph rewriting, where we model message passing over reliable and unreliable channels.

The article is structured as follows: First, in Section 2 we recite the fundamental ideas for reactive systems without conditions, including all preliminary definitions and techniques developed for reactive systems relevant to our work. In Section 3, we consider the refinement to conditional reactive systems, before we turn towards our main contribution in Section 4, which is conditional bisimulation and its up-to variant in Section 5. In Section 6 we give an alternative characterization of conditional bisimilarity and compare to related notions of behavioural equivalence and we conclude in Section 7. All proofs for the theorems in Sections 4 to 6, as well as additional examples can be found in the full version [17].

2 Reactive Systems

2.1 Reactive Systems without Conditions

We denote the composition of arrows $f\colon A \to B,\ g\colon B \to C$ by $f;g\colon A \to C$.

We now define reactive systems, introduced in [22] and extended in [16] with application conditions for rules:

▶ **Definition 2.1** (Reactive system rules, reaction)**.** *Let* $\mathbf{C}$ *be a category with a distinguished object* 0 *(not necessarily initial). A* rule *is a pair* (ℓ, r) *of arrows* $\ell, r\colon 0 \to I$ *(called left-hand side and right-hand side). A* reactive system *is a set of rules.*

Let $\mathcal{R}$ *be a reactive system and* $a, a'\colon 0 \to J$ *be arrows. We say that* a reduces to a' *(*$a \rightsquigarrow a'$*) whenever there exists a rule* $(\ell, r) \in \mathcal{R}$ *with* $\ell, r\colon 0 \to I$ *and an arrow* $c\colon I \to J$ *(the* reactive context*) such that* $a = \ell;c$ *and* $a' = r;c$*.*

Using a notation closer to process caluli, we could write $C[P] \rightsquigarrow C[P']$ whenever there is a reaction rule $P \to P'$ and a context $C[\_]$. Fixing a distinguished object 0 means that we consider only ground reaction rules (as opposed to open reactive systems [19]).

An important instance are reactive systems where the arrows are cospans in a base category $\mathbf{D}$ with pushouts [27, 28]. A *cospan* is a pair of arrows $f_L\colon A \to C,\ f_R\colon B \to C$. A cospan is *input linear* if its left arrow f_L is mono.

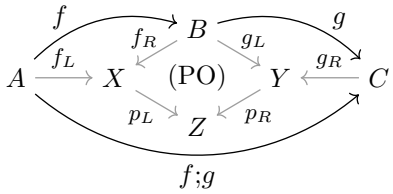


Figure 1 Composition of cospans via pushouts.

Two cospans $f\colon A \xrightarrow{f_L} X \xleftarrow{f_R} B,\ g\colon B \xrightarrow{g_L} Y \xleftarrow{g_R} C$ are composed by taking the pushout (p_L, p_R) of (f_R, g_L) as shown in Figure 1. The result is the cospan $f;g\colon A \xrightarrow{f_L;p_L} Z \xleftarrow{g_R;p_R} C$, where Z is the pushout object of $f_R,\ g_L$. For adhesive categories [20], the composition of input linear cospans again yields an input linear cospan (by applying [20, Lemma 12] to the cospan composition diagram). Given an adhesive category $\mathbf{D}$, $ILC(\mathbf{D})$ is the category where the objects are the objects of $\mathbf{D}$, the arrows $f\colon A \to C$ are input linear cospans $f\colon A \to B \leftarrow C$ of $\mathbf{D}$ and composition is performed via pushouts as above. We see an arrow $f\colon A \to C$ of $ILC(\mathbf{D})$ as an object B of $\mathbf{D}$ equipped with two interfaces A, C, and composition glues the inner objects of two cospans via their interfaces. Input linearity is required since we rely on adhesive categories where pushouts along monos are well-behaved and are stable under pullbacks.

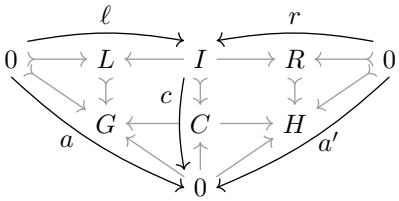


Figure 2 DPO graph transformation as reactive system steps.

In this article, as a running example we consider $\mathbf{Graph_{fin}}$, which is the category of finite graphs (we use directed multigraphs with node and edge labels) and total graph morphisms as arrows. In $\mathbf{Graph_{fin}}$, monos are exactly the injective graph morphisms. We then use reactive systems over $ILC(\mathbf{Graph_{fin}})$ (input-linear cospans of graphs), i.e. we rewrite graphs with interfaces. If the distinguished object 0 is the empty graph (the initial object of $\mathbf{Graph_{fin}}$), such reactive systems coincide [27] with the well-known *double pushout (DPO) graph transformation* approach [10, 13] when used with injective matches. As shown in Figure 2, a DPO rewrite step $G \Rightarrow H$ can be expressed as a reactive system reaction $a \rightsquigarrow a'$ where the pushouts of the DPO step are obtained from cospan compositions $\ell;c$ and $r;c$.

2.2 Deriving Bisimulation Congruences

The reduction relation $\rightsquigarrow$ generates an unlabelled transition system, on reactive agents (in our example, graphs) as states. A disadvantage of bisimilarity on $\rightsquigarrow$ is that it usually is not a congruence: it is easy to construct an example where neither a nor b can perform a

step since no complete left-hand side is present. However, by adding a suitable context c, $a;c$ could contain a full left-hand side and can reduce, whereas $b;c$ can not.

Therefore, to check whether two components can be exchanged, they have to be combined with every possible context and bisimilarity has to be shown for each.

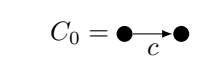
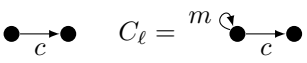
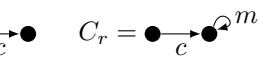
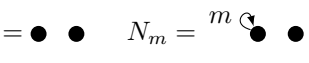
In order to obtain a congruence, we can resort to defining bisimulation on labelled transitions, using as labels the additional contexts that allow an agent to react [22, 16].

▶ **Definition 2.2** (Context step (without conditions) [16]). *Let $\mathcal{R}$ be a reactive system and $a\colon 0 \to J$, $f\colon J \to K$, $a'\colon 0 \to K$ be arrows. We write $a \xrightarrow{f}_C a'$ whenever $a;f \rightsquigarrow a'$ (i.e. there exists a rule $(\ell, r) \in \mathcal{R}$ and an arrow c such that $a;f = \ell;c$, $a' = r;c$). Such steps are called* context steps.

$$\begin{array}{ccccc} 0 & \xrightarrow{\ell} & I & \xleftarrow{r} & 0 \\ {\scriptstyle a}\downarrow & & {\scriptstyle c}\downarrow & \swarrow_{a'} & \\ J & \xrightarrow{f} & K & & \end{array}$$

The name *context step* stems from the fact that a cannot do a reaction on its own, but requires an additional context f. This can be seen in the following example:

▶ **Example 2.3** (Context step (without conditions)). Consider the following reactive system over $ILC(\mathbf{Graph_{fin}})$, where we model a network of nodes that pass messages (represented by m-loops) over communication channels. Let the following graphs be given:

$C_0 =$  $C_\ell =$  $C_r =$  $N_0 =$  $N_m =$ 

We can now represent the transmission of a message from the left node to the right node using the rule $P = (\emptyset \to C_\ell \leftarrow C_0,\ \emptyset \to C_r \leftarrow C_0)$. All graph morphisms are induced by edge labels and position of nodes, i.e. the left node is always mapped to the left node.

Observe that a channel by itself ($a = \emptyset \to C_0 \leftarrow N_0$) cannot do a reaction, since there is no message to be transferred. However, if a message on the left node is borrowed ($f = N_0 \to N_m \leftarrow N_0$), the example rule can be applied. As a result, we obtain the context step $(\emptyset \to C_0 \leftarrow N_0) \xrightarrow{(N_0 \to N_m \leftarrow N_0)}_C (\emptyset \to C_r \leftarrow N_0)$.

A bisimulation relation over $\to_C$ is called *saturated bisimulation*, as it checks all contexts. Consequently, saturated bisimilarity $\sim_C$ ($\sim_{SAT}$ in [16]) is a congruence [5, 16], i.e., it is closed under contextualization. In other words $a \sim_C b$ implies $a;c \sim_C b;c$ for all contexts c.

2.3 Representative Squares

Checking bisimilarity of context steps is impractical: usually, f can be chosen from an infinite set of possible contexts, which all have to be checked. Most of these contexts are larger than necessary, that is, they contain elements that do not actively participate in the reduction. (In Example 2.3, contexts can be arbitrarily large, as long as they have an m-loop on the left node.) An improvement would be to check only the minimal contexts from which all other context steps can be derived.

When checking which contexts are required to make a rule applicable, in the reaction diagram (Definition 2.2) the arrows a, ℓ are given and we need to check for possible values of f (which generate matching c, a'). To derive a set of contexts f which is as small as possible – preferably finite – [6, 16] introduced the notion of representative squares, which describe methods to produce squares from a pair a, ℓ in a representative way.

▶ **Definition 2.4** (Representative squares [6]). *A class κ of commuting squares in a category $\mathbf{C}$ is* representative *if κ satisfies the following condition: for each commuting square $(\alpha_1, \alpha_2, \delta_1, \delta_2)$ in $\mathbf{C}$ there exists a commuting square $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ in κ and an arrow γ, such that $\delta_1 = \beta_1;\gamma$, $\delta_2 = \beta_2;\gamma$. This situation is depicted in Figure 3.*

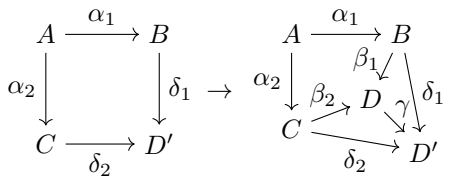


Figure 3 Every commuting square of the category (left) can be reduced to a representative square in κ and an arrow γ which extends the representative square to the original square (right).

For two arrows $\alpha_1 \colon A \to B$, $\alpha_2 \colon A \to C$, we define $\kappa(\alpha_1, \alpha_2)$ as the set of pairs of arrows (β_1, β_2) which, together with α_1, α_2, form representative squares in κ.

The original paper on reactive systems [22] used the (more restrictive) notion of idem pushouts instead of representative squares. Unfortunately, the universal property of idem pushouts leads to complications, in particular for cospan categories, where one has to resort to the theory of bicategories in order to be able to express this requirement. For the purposes of this paper, we stick to the simpler notion of representative squares, in order to keep our results independent of the concrete class of squares chosen.

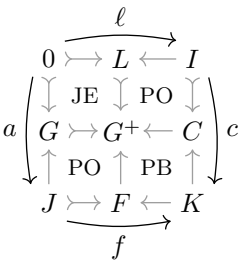
The question arises which constructions yield suitable classes of representative squares, ideally with finite $\kappa(\alpha_1, \alpha_2)$, in order to represent all possible contexts δ_1, δ_2 with a finite set of representative contexts β_1, β_2. Pushouts can be used when they exist [16], however, they do not exist for $ILC(\mathbf{Graph_{fin}})$.

For adhesive categories, borrowed context diagrams – initially introduced as an extension of DPO rewriting [9] – can be used as representative squares. Before we can introduce such diagrams, we first need the notion of jointly epi.

▶ **Definition 2.5** (Jointly epi). *A pair of arrows $f \colon B \to D$, $g \colon C \to D$ is* jointly epi *(*JE*) if for each pair of arrows $d_1, d_2 \colon D \to E$ the following holds: if $f;d_1 = f;d_2$ and $g;d_1 = g;d_2$, then $d_1 = d_2$.*

In $\mathbf{Graph_{fin}}$ jointly epi equals jointly surjective, meaning that each node or edge of D is required to have a preimage under f or g or both (it contains only elements from B or C).

▶ **Definition 2.6** (Borrowed context diagram [16]). *A commuting diagram in the category $ILC(\mathbf{C})$, where $\mathbf{C}$ is adhesive, is a* borrowed context diagram *whenever it has the form of the diagram shown below, and the four squares in the base category $\mathbf{C}$ are jointly epi (JE), pushout (PO) or pullback (PB) as indicated.*



The top left jointly epi square and the bottom left pushout ensure that the borrowed context f is not larger than necessary [9]. We will discuss an example below (Example 2.9). For additional examples, we refer to [9].

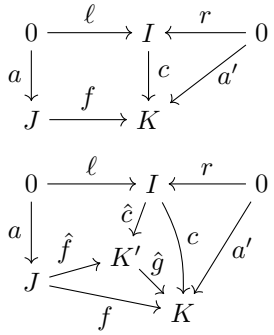
For adhesive categories, borrowed context diagrams form a representative class of squares [16]. Furthermore, for some categories (such as $\mathbf{Graph_{fin}}$), there are – up to isomorphism – only finitely many jointly epi squares for a given span of monos and hence only finitely many borrowed context diagrams given a, ℓ (since pushout complements along monos in adhesive categories are unique up to isomorphism).

This motivates the following finiteness assumption that we will refer to in this paper: given a, ℓ, we require that $\kappa(a, \ell)$ is finite. (FIN)

2.4 Representative Steps

It is possible to define a reaction relation based on representative squares. By requiring that the left square is representative, we ensure that the contexts $\hat{f}$ are not larger than necessary:

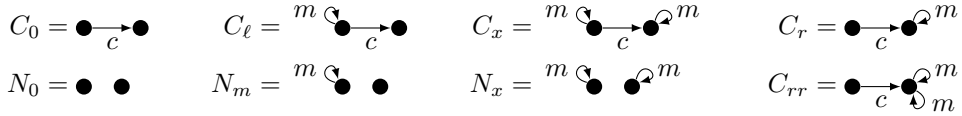
▶ **Definition 2.7** (Representative step (without conditions) [16]). *Let $a\colon 0 \to J$, $\hat{f}\colon J \to K$, $a'\colon 0 \to K$ be arrows. We write $a \xrightarrow{\hat{f}}_R a'$ if a context step $a \xrightarrow{\hat{f}}_C a'$ is possible (i.e. $a;\hat{f} \rightsquigarrow a'$, i.e. for some rule (ℓ, r) and some arrow $\hat{c}$ we have $a;\hat{f} = \ell;\hat{c}$ and $r;\hat{c} = a'$) and additionally $\kappa(a, \ell) \ni (\hat{f}, \hat{c})$ (i.e. the arrows $(a, \ell, \hat{f}, \hat{c})$ form a representative square). Such steps are called* representative steps.



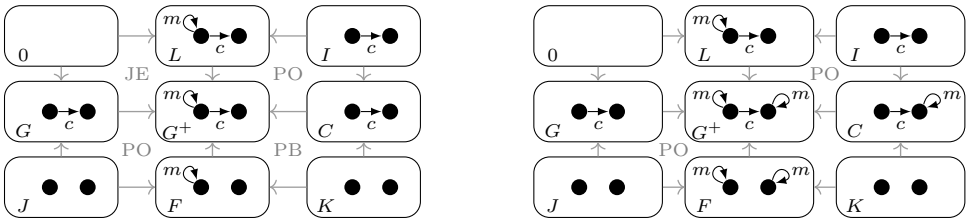
▶ Remark 2.8. Definitions 2.2 and 2.4 imply that every context step $a \xrightarrow{f}_C a'$ (top diagram) can be reduced to a representative step $a \xrightarrow{\hat{f}}_R r;\hat{c}$ (bottom diagram), a fact used in the proofs.

For this, we construct the representative square $(a, \ell, \hat{f}, \hat{c}) \in \kappa$ (which, according to Definition 2.4, always exists) from the square (a, ℓ, f, c) describing the context step. We obtain arrows $\hat{f}, \hat{c}$ and an arrow $\hat{g}$ which completes $\hat{f}, \hat{c}$ to f, c (i.e. $\hat{f};\hat{g} = f,\ \hat{c};\hat{g} = c$).

▶ **Example 2.9** (Representative steps). Let the following graphs be given:



As before (Example 2.3), the rule $P = (\emptyset \to C_\ell \leftarrow C_0,\ \emptyset \to C_r \leftarrow C_0)$ transfers a message. One possible context step allows a channel C_0 to borrow a message N_m and do a transfer: $(\emptyset \to C_0 \leftarrow N_0) \xrightarrow{(N_0 \to N_m \leftarrow N_0)}_C (\emptyset \to C_r \leftarrow N_0)$.



(a) Borrowed context diagram for $C_0 \xrightarrow{N_m}_R C_r$. **(b)** Commuting diagram for $C_0 \xrightarrow{N_x}_C C_{rr}$.

Figure 4 Diagrams for the two steps described in Example 2.9.

Another possible context step is $(\emptyset \to C_0 \leftarrow N_0) \xrightarrow{(N_0 \to N_x \leftarrow N_0)}_C (\emptyset \to C_{rr} \leftarrow N_0)$, i.e. an additional message on the right node is borrowed. Clearly, this is a valid context step, but the right message is not required by the rule, and we do not want to consider such steps in our analysis (by adding yet more messages, we obtain infinitely many context steps).

However, the second context step is not a representative step. We try to construct a borrowed context diagram: First we fill in the graphs given by a, f and ℓ, then we construct the bottom left pushout, we obtain $G^+ = C_x$ as depicted in Figure 4b. Then however the top left square is not jointly epi, since neither C_ℓ (from ℓ) nor C_0 (from a) provide a preimage for the right m-loop.

On the other hand, the first context step is representative, since there $G^+ = C_\ell$ does not contain the problematic right m-loop and it is possible to complete the borrowed context diagram as shown in Figure 4a. (To obtain the result of the step, the right-hand side a' is constructed just as for context steps (see Example 2.3), which is not depicted here.)

In a *semi-saturated bisimulation*, $\to_R$-steps are answered by $\to_C$-steps (for every $(a, b) \in R$ and step $a \xrightarrow{f}_R a'$ there is $b \xrightarrow{f}_C b'$ such that $(a', b') \in R$). The resulting bisimilarity $\sim_R$ is identical [16] to saturated bisimilarity (i.e. $\sim_R = \sim_C$) and therefore also a congruence. Whenever (Fin) holds, $\sim_R$ is amenable to mechanization, since we have to consider only finitely many $\to_R$-steps ($\to_R$ is finitely branching).

Note that answering $\to_R$-steps with $\to_R$-steps gives a different, finer notion of behavioural equivalence, which we do not treat here [16].

3 Conditions for Reactive Systems

The reactive systems defined so far cannot represent rules where a certain component is required to be absent: whenever a reaction $a \rightsquigarrow a'$ is possible, a reaction $a;c \rightsquigarrow a';c$ (with additional context c) is also possible, with no method to prevent this. Restricting rule applications can be useful, e.g. to model access to a shared resource, which may only be accessed if no other entity is currently using it.

For graph transformation systems, application conditions with a first-order logic flavour have been studied extensively (e.g. in [12, 14]) and generalized to reactive systems in [6]. If we interpret such conditions in $ILC(\mathbf{Graph_{fin}})$, we obtain a logic that subsumes first-order logic (for more details on expressiveness see [6]).

In this section, we summarize the definitions from [6] and define shifting of conditions as partial evaluation. We then summarize the changes that are necessary to extend reactive systems with conditions. An example for conditional reactive systems will be discussed later (Example 4.3). For further examples, we refer to the full version and to [6].

3.1 Conditions and Satisfiability

▶ **Definition 3.1** (Condition [6]). *Let* $\mathbf{C}$ *be a category. The set of conditions* $\mathrm{Cond}(A)$ *over an object* A *is defined inductively as:*

- $\mathrm{true}_A := (A, \forall, \emptyset) \in \mathrm{Cond}(A)$, $\mathrm{false}_A := (A, \exists, \emptyset) \in \mathrm{Cond}(A)$ *(base case)*
- $\mathcal{A} = (A, \mathcal{Q}, S) \in \mathrm{Cond}(A)$, *where* $A = \mathrm{Ro}(\mathcal{A})$ *is the* root object *of* $\mathcal{A}$,
  - $\mathcal{Q} \in \{\forall, \exists\}$ *is a quantifier and*
  - S *is a finite set of pairs* $(h, \mathcal{A}')$, *where* $h\colon A \to A'$ *is an arrow and* $\mathcal{A}' \in \mathrm{Cond}(A')$.

Note that conditions can be represented as finite trees.

▶ **Definition 3.2** (Satisfiability of conditions [6]). *Let* $\mathcal{A} \in \mathrm{Cond}(A)$. *For an arrow* $a\colon A \to B$ *and a condition* $\mathcal{A}$ *we define the satisfaction relation* $a \models \mathcal{A}$ *as follows:*

- $a \models (A, \forall, S)$ *iff for every pair* $(h, \mathcal{A}') \in S$ *and every arrow* $g\colon \mathrm{Ro}(\mathcal{A}') \to B$ *we have: if* $a = h;g$, *then* $g \models \mathcal{A}'$.
- $a \models (A, \exists, S)$ *iff there exists a pair* $(h, \mathcal{A}') \in S$ *and an arrow* $g\colon \mathrm{Ro}(\mathcal{A}') \to B$ *such that* $a = h;g$ *and* $g \models \mathcal{A}'$.

We write $\mathcal{A} \models \mathcal{B}$ *(*$\mathcal{A}$ *implies* $\mathcal{B}$*) if for every arrow* c *with* $\mathrm{dom}(c) = \mathrm{Ro}(\mathcal{A}) = \mathrm{Ro}(\mathcal{B})$ *we have: if* $c \models \mathcal{A}$, *then* $c \models \mathcal{B}$. *Two conditions are equivalent (*$\mathcal{A} \equiv \mathcal{B}$*) if* $\mathcal{A} \models \mathcal{B}$ *and* $\mathcal{B} \models \mathcal{A}$.

▶ **Proposition 3.3** (Boolean operations). *We define the following Boolean operations on conditions:*

- $\neg(A, \forall, S) := (A, \exists, \{(h, \neg\mathcal{A}') \mid (h, \mathcal{A}') \in S\})$ *and*
  $\neg(A, \exists, S) := (A, \forall, \{(h, \neg\mathcal{A}') \mid (h, \mathcal{A}') \in S\})$
- $\mathcal{A} \vee \mathcal{B} := (A, \exists, \{(\mathrm{id}_A, \mathcal{A}), (\mathrm{id}_A, \mathcal{B})\})$ *for two conditions* $\mathcal{A}, \mathcal{B} \in \mathrm{Cond}(A)$
- $\mathcal{A} \wedge \mathcal{B} := (A, \forall, \{(\mathrm{id}_A, \mathcal{A}), (\mathrm{id}_A, \mathcal{B})\})$ *for two conditions* $\mathcal{A}, \mathcal{B} \in \mathrm{Cond}(A)$

These operations satisfy the standard laws of propositional logic, i.e. true_A *is satisfied by every arrow with domain* A, false_A *is satisfied by no arrow;* $a \models \neg\mathcal{A}$ *if and only if* $a \not\models \mathcal{A}$; $a \models (\mathcal{A} \vee \mathcal{B})$ *if and only if* $a \models \mathcal{A} \vee a \models \mathcal{B}$, *analogously for* $\mathcal{A} \wedge \mathcal{B}$.

3.2 Shifting as Partial Evaluation of Conditions

When evaluating conditions, it is sometimes known that a given context is guaranteed to be present. In this case, a condition can be rewritten, using representative squares, under the assumption that this context is provided by the environment. This operation is known as shift [14]:

▶ **Definition 3.4** (Shift of a condition [6]). *Given a fixed class of representative squares* κ, *the* shift of a condition $\mathcal{A} = (A, \mathcal{Q}, S)$ along an arrow $c\colon A \to B$ *is inductively defined as follows:*

$$\mathcal{A}_{\downarrow c} := \left(B, \mathcal{Q}, \left\{(\beta, \mathcal{A}'_{\downarrow\alpha}) \;\middle|\; (h, \mathcal{A}') \in S,\ (\alpha, \beta) \in \kappa(h, c)\right\}\right)$$

The shift operation can be understood as a partial evaluation of $\mathcal{A}$ *under the assumption that* c *is already present. It satisfies* $c;d \models \mathcal{A} \iff d \models \mathcal{A}_{\downarrow c}$.

If we assume that (Fin) holds, shifting a finite condition will again result in a finite condition. Representative squares as well as shift play a major role in the diagrammatic proofs.

3.3 Conditional Reactive Systems

We now extend reactive systems with application conditions:

▶ **Definition 3.5** (Conditional reactive system [6]). *A* rule with condition *is a triple* $(\ell, r, \mathcal{B})$ *where* $\ell, r\colon 0 \to I$ *are arrows and* $\mathcal{B}$ *is a condition with root object* I*. A* conditional reactive system *is a set of rules with conditions.*

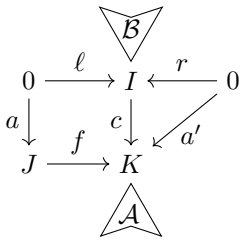
As the root object I of the condition is the codomain of the rule arrow, it is also the domain of the reactive context, which has to satisfy the rule condition in order to be able to apply the rule:

▶ **Definition 3.6** (Reaction). *Let* a, a' *be arrows of a conditional reactive system with rules* $\mathcal{R}$*. We say that* a reduces to a' *(*$a \rightsquigarrow a'$*) whenever there exists a rule* $(\ell, r, \mathcal{B}) \in \mathcal{R}$ *with* $\ell, r\colon 0 \to I$ *and a reactive context* $c\colon I \to J$ *such that* $a = \ell;c$*,* $a' = r;c$ *and additionally* $c \models \mathcal{B}$*.*

In order to define a bisimulation for conditional reactive systems that is also a congruence, it is necessary to enrich labels with conditions derived from the application conditions. Since we can not assume that the full context is present, the application condition might refer to currently unknown parts of the context and this has to be suitably integrated into the label.

▶ **Definition 3.7** (Context/representative step with conditions [16]). *Let* $\mathcal{R}$ *be a conditional reactive system, let* $a\colon 0 \to J$*,* $f\colon J \to K$*,* $a'\colon 0 \to K$ *be arrows and* $\mathcal{A} \in \mathrm{Cond}(K)$ *be a condition. We write* $a \xrightarrow{f,\mathcal{A}}_C a'$ *whenever there exists a rule* $(\ell, r, \mathcal{B}) \in \mathcal{R}$ *and an arrow* c *such that* $a;f = \ell;c$*,* $a' = r;c$ *(i.e. the reaction is possible without conditions) and furthermore* $\mathcal{A} \models \mathcal{B}_{\downarrow c}$ *(an additional context has to satisfy a condition* $\mathcal{A}$ *which is at least as strong as the rule condition* $\mathcal{B}$*, shifted over* c*). Such steps are called* context steps*.*

We write $a \xrightarrow{f,\mathcal{A}}_R a'$ *whenever* $a \xrightarrow{f,\mathcal{A}}_C a'$*,* $\kappa(a, \ell) \ni (f, c)$ *and* $\mathcal{A} = \mathcal{B}_{\downarrow c}$*. Such steps are called* representative steps*.*



Conditions are represented graphically in the form of "arrowhead shapes" depicted next to the root object. Intuitively $a \xrightarrow{f,\mathcal{A}}_C a'$ means that a can make a step to a' when borrowing f, if the yet unknown context beyond f satisfies condition $\mathcal{A}$ (since this context does not directly participate in the reduction, we call it *passive context*). In the case of a representative step, we require that a context step is possible, the borrowed context is minimal, and the condition on the passive context is not stronger than necessary.

▶ Remark 3.8. Definitions 2.4 and 3.7 imply, analogously to Remark 2.8, that every context step $a \xrightarrow{f,\mathcal{A}}_C a'$ can be reduced to a representative step $a \xrightarrow{\hat{f},\,\mathcal{B}_{\downarrow \hat{c}}}_R r;\hat{c}$.

We now extend (semi-)saturated bisimilarity to rules with conditions:

▶ **Definition 3.9** ((Semi-)Saturated bisimilarity [16]). *A* saturated bisimulation *is a symmetric relation* R*, relating pairs of arrows* $a, b\colon 0 \to J$*, such that: for all* $(a, b) \in R$ *and for every context step* $a \xrightarrow{f,\mathcal{A}}_C a'$ *there exist answering moves* $b \xrightarrow{f,\mathcal{B}_i}_C b'_i$*,* $i \in I$*, such that* $(a', b'_i) \in R$ *and* $\mathcal{A} \models \bigvee_{i \in I} \mathcal{B}_i$*.*

Two arrows a, b are called saturated bisimilar *($(a,b) \in \sim_C$) whenever there exists a saturated bisimulation R with $(a,b) \in R$. Similarly, for* semi-saturated bisimilarity *we require that $\rightarrow_R$-steps of a can be answered by $\rightarrow_C$-steps of b. Saturated and semi-saturated bisimilarity agree and both are congruences [16].*

The logic does not support infinite disjunctions, so $\mathcal{A} \models \bigvee_{i \in I} \mathcal{B}_i$ means that for every d with $d \models \mathcal{A}$, there exists $i \in I$ such that $d \models \mathcal{B}_i$.

4 Conditional Bisimilarity

We will now introduce our new results on conditional bisimilarity: as stated earlier, our motivation is to extend the notion of saturated bisimilarity, which is often too strict, since it requires that two system states behave identically in all possible contexts. However, sometimes it is enough to ensure behavioural equivalence only in specific environments.

Hence we now replace standard bisimilarity, which is a binary relation, by a ternary relation – called conditional relation – with tuples of the form $(a, b, \mathcal{C})$, which can be read as: a, b are bisimilar in all contexts satisfying $\mathcal{C}$.

4.1 Definition, Properties and Examples

▶ **Definition 4.1** (Conditional relation, closure under contextualization, conditional congruence)**.** *A* conditional relation *is a set of triples $(a, b, \mathcal{C})$, where $a, b \colon 0 \to J$ are arrows with identical target and $\mathcal{C}$ is a condition over J. A conditional relation R is* reflexive *if $(a, a, \mathcal{C}) \in R$ for all $a, \mathcal{C}$ with $\mathrm{codom}(a) = \mathrm{Ro}(\mathcal{C})$;* symmetric *if $(a, b, \mathcal{C}) \in R$ implies $(b, a, \mathcal{C}) \in R$;* transitive *if $(a, b, \mathcal{C}) \in R$ and $(b, c, \mathcal{C}) \in R$ implies $(a, c, \mathcal{C}) \in R$. R is* closed under contextualization *if $(a, b, \mathcal{C}) \in R$ implies $(a;d, b;d, \mathcal{C}_{\downarrow d}) \in R$. R is a* conditional congruence *if it is additionally an equivalence (reflexive, symmetric, transitive).*

Closure under contextualization means that whenever a, b are related under a context satisfying $\mathcal{C}$, then they are still related when we contextualize under d, where however the condition has to be shifted since we commit to the fact that the context is of the form $d;c$ for some c.

Note that the root object of the condition is not the source of a (as is the case for satisfiability), but the target $\mathrm{codom}(a)$. This is because we do not state a condition on the arrows a, b themselves, but on the context in which they are embedded ($a;f$ resp. $b;f$ for some context f), so the condition is over $\mathrm{dom}(f) = \mathrm{codom}(a)$.

▶ **Definition 4.2** (Conditional bisimulation)**.** *A conditional bisimulation R is a symmetric conditional relation such that the following holds: for each triple $(a, b, \mathcal{C}) \in R$ and each context step $a \xrightarrow{f, \mathcal{A}}_C a'$, there are answering steps $b \xrightarrow{f, \mathcal{B}_i}_C b'_i$, $i \in I$, and conditions $\mathcal{C}'_i$ such that $(a', b'_i, \mathcal{C}'_i) \in R$ and $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$. Two arrows are* conditionally bisimilar under $\mathcal{C}$ *($(a, b, \mathcal{C}) \in \mathrel{\mathring{\sim}}_C$) whenever a conditional bisimulation R with $(a, b, \mathcal{C}) \in R$ exists.*[1]

The condition is to be understood as follows: For every step, we have a borrowed context f and an additional passive context d (as explained below Definition 3.7). The condition $\mathcal{C}$ from the triple refers to the full context of a (hence $f;d \models \mathcal{C}$ or equivalently $d \models \mathcal{C}_{\downarrow f}$), while $\mathcal{A}$, coming from the context step, only refers to the passive context (hence $d \models \mathcal{A}$).

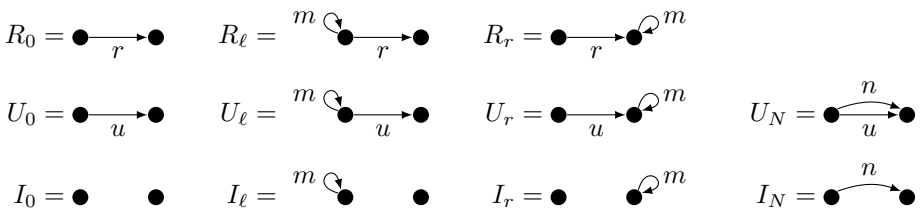
[1] Note that since conditional bisimulations are closed under union, $\mathrel{\mathring{\sim}}_C$ is itself a conditional bisimulation.

If these two are satisfied (left-hand side of the implication), we require answering steps which also impose conditions on the passive context ($d \models \mathcal{B}_i$). Additionally, we choose conditions $\mathcal{C}'_i$, which ensure that the chosen answering steps yield pairs a', b'_i which are bisimilar under $\mathcal{C}'_i$. As for saturated bisimilarity [16, remark after Definition 15], we need to allow several answering moves for a single step of a: the answering step taken by b might depend on the context, using different rules for contexts satisfying different conditions $\mathcal{B}_i$. We just have to ensure that all answering step conditions together (disjunction on the right-hand side) fully cover the conditions under which the step of a is feasible (left-hand side).

▶ **Example 4.3** (Message passing over unreliable channels)**.** We now work in the category of input-linear cospans of graphs, i.e., $ILC(\mathbf{Graph_{fin}})$.

We extend our previous example (cf. Example 2.3) of networked nodes, introducing different types of channels. A channel can be reliable or unreliable, indicated by an r-edge or u-edge respectively. Sending a message over a reliable channel always succeeds (rule P_R), while an unreliable channel only transmits a message if there is no noise (indicated by a parallel n-edge) in the environment that disturbs the transmission (rule P_U).

To represent this situation as a reactive system, let the following graphs be given:



We can now represent the transmission of a message using the following rules with application conditions, where $\mathcal{A}_U$ states that no n-edge exists:

$$P_R = \left(\emptyset \to R_\ell \leftarrow R_0,\ \emptyset \to R_r \leftarrow R_0,\ \mathrm{true}_{R_0}\right)$$
$$P_U = \left(\emptyset \to U_\ell \leftarrow U_0,\ \emptyset \to U_r \leftarrow U_0,\ \mathcal{A}_U\right)$$
$$\mathcal{A}_U = \left(U_0, \forall, \left\{(U_0 \to U_N \leftarrow U_0,\ \mathrm{false}_{U_0})\right\}\right)$$

Hence the application condition $\mathcal{A}_U$ says that the context must not be decomposable into $U_0 \to U_N \leftarrow U_0$ and some other cospan, i.e., the u-edge in the interface has no parallel n-edge. In other words: there is no noise.

We compare the behaviour of a reliable channel ($r := \emptyset \to R_0 \leftarrow I_0$) to that of an unreliable channel ($u := \emptyset \to U_0 \leftarrow I_0$). It is easy to see that they are not saturated bisimilar: r can do a step by borrowing a message on the left ($f := I_0 \to I_\ell \leftarrow I_0$) without further restrictions (i.e. using an environment condition $\mathcal{A} = \mathrm{true}$). But u is unable to answer this step, because the corresponding rule is only applicable if no n-edge is present.

However, r and u are conditionally bisimilar under the assumption that no n-edge is present ($\mathcal{C} = \mathcal{A}_C = \left(I_0, \forall, \left\{(I_0 \to I_N \leftarrow I_0,\ \mathrm{false}_{I_0})\right\}\right)$), i.e. there exists a conditional bisimulation that contains $(r, u, \mathcal{A}_C)$. A direct proof is hard, since the proof involves checking infinitely many context steps, since messages accumulate on the right-hand side. However, in Example 4.9 we will use an argument based on representative steps to construct a proof.

▶ Remark 4.4 (Condition strengthening). It holds that $(a, b, \mathcal{C}') \in \mathring{\sim}_C$, $\mathcal{C} \models \mathcal{C}'$ implies $(a, b, \mathcal{C}) \in \mathring{\sim}_C$. (This is due to the fact that $\mathcal{C} \models \mathcal{C}'$ implies $\mathcal{C}_{\downarrow f} \models \mathcal{C}'_{\downarrow f}$ which, in Definition 4.2, implies $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \mathcal{A} \wedge \mathcal{C}'_{\downarrow f}$ for any condition $\mathcal{A}$ and arrow f.)

▶ Remark 4.5. It can be shown that conditional bisimilarity $\overset{\circ}{\sim}_C$ is a conditional congruence, this follows as a corollary of Theorem 6.3 which will be shown later. This is an important plausibility check, since reactive systems have been introduced with the express purpose to define and reason about bisimulation congruences.

Our motivation for introducing the notion of conditional bisimilarity was to check whether two systems are behaviourally equivalent, when they are put into a context that satisfies some condition $\mathcal{C}$. It is not immediately obvious that our definition can be used for this purpose, since all context steps are checked, not just the ones that actually satisfy $\mathcal{C}$.

Hence we now show that our definition is sound, i.e. if two systems are conditionally bisimilar, then they show identical behaviour under all contexts that satisfy $\mathcal{C}$.

▶ **Theorem 4.6.** *Let R be a conditional bisimulation. Then $R' = \{(a;d, b;d) \mid (a, b, \mathcal{C}) \in R \wedge d \models \mathcal{C}\}$ is a bisimulation for the reaction relation $\rightsquigarrow$.*

Note that the converse of Theorem 4.6 (if R' is a bisimulation, then R is a conditional bisimulation) does not hold. For a counterexample, we refer to the full version.

4.2 Representative Conditional Bisimulations

Checking whether two arrows are conditionally bisimilar, or whether a given relation is a conditional bisimulation, can be hard in practice, since we have to check all possible context steps, of which there are typically infinitely many.

For saturated bisimilarity, we used representative steps instead of context steps (cf. Sections 2.3 and 2.4) to reduce the number of contexts to be checked. In this section, we extend our definition of conditional bisimulation to use representative steps and prove that the resulting bisimilarity is identical to the one previously defined.

▶ **Definition 4.7** (Representative conditional bisimulation)**.** *A* representative conditional bisimulation *R is a symmetric conditional relation such that the following holds: for each triple $(a, b, \mathcal{C}) \in R$ and each representative step $a \xrightarrow{f, \mathcal{A}}_R a'$, there are answering context steps $b \xrightarrow{f, \mathcal{B}_i}_C b'_i$ and conditions $\mathcal{C}'_i$ such that $(a', b'_i, \mathcal{C}'_i) \in R$ and $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$. Two arrows are* representative conditionally bisimilar under $\mathcal{C}$ *($(a, b, \mathcal{C}) \in \overset{\circ}{\sim}_R$) whenever a representative conditional bisimulation R with $(a, b, \mathcal{C}) \in R$ exists.*

We now show that these two conditional bisimilarities are equivalent.

▶ **Theorem 4.8.** *Conditional bisimilarity and representative conditional bisimilarity coincide, that is, $\overset{\circ}{\sim}_C = \overset{\circ}{\sim}_R$.*

▶ **Example 4.9** (Message passing over unreliable channels, continued)**.** Consider the reactive system of Example 4.3. There exists a representative conditional bisimulation R such that $(\emptyset \to R_0 \leftarrow I_0,\ \emptyset \to U_0 \leftarrow I_0,\ \mathcal{A}_C) \in R$.

We consider the representative steps that are possible from either R_0 or U_0 and only explain the most interesting cases:

- R_0 can do a step using rule P_R by borrowing a message on the left node, that is, $f = I_0 \to I_\ell \leftarrow I_0$, and reacting to R_r. No further restrictions on the environment are necessary, so $\mathcal{A} = \text{true}$. U_0 can answer this step using P_U and reacts to U_r, but only if no noise is present (environment satisfies $\mathcal{B}_i = \mathcal{A}_C$). We evaluate the implication $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \equiv \text{true} \wedge \mathcal{A}_{C\downarrow f} \equiv \mathcal{A}_C \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{A}_C) \equiv \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$, setting $\mathcal{C}'_i = \mathcal{A}_C$. (Note that $\mathcal{A}_{C\downarrow f} \equiv \mathcal{A}_C$ since $\mathcal{A}_C$ forbids the existence of an n-edge between the two interface nodes and f is unrelated, providing an m-loop on the left-hand node.) We now require $(\emptyset \to R_r \leftarrow I_0,\ \emptyset \to U_r \leftarrow I_0,\ \mathcal{A}_C) \in R$.

- Symmetrically, U_0 can do a step using P_U by borrowing a message on the left node, reacting to U_r in an environment without noise ($\mathcal{A} = \mathcal{A}_C$). R_0 can answer this step under any condition $\mathcal{B}_i$. Then, the implication is satisfied if we set $\mathcal{C}'_i = \mathcal{A}_C$, so we require again $(\emptyset \to R_r \leftarrow I_0,\ \emptyset \to U_r \leftarrow I_0,\ \mathcal{A}_C) \in R$.
- There are additional representative steps that differ in how much of the left-hand side is borrowed, but can be proven analogously to the two previously discussed steps.

This means that we have to add the pair $(\emptyset \to R_r \leftarrow I_0,\ \emptyset \to U_r \leftarrow I_0,\ \mathcal{A}_C)$ to R and to continue adding pairs until we obtain a bisimulation: with every step, a new triple with an additional m-loop on the right node is added to the relation, therefore, the smallest conditional bisimulation has infinite size. However, except for the additional m-loop on the right node, which does not affect rule application, this pair is identical to the initial one and we can hence use a similar argument. In Section 5 we show how to make this formal, using up-to techniques, and thus obtain a completely mechanized proof. In summary, we conclude that R_0 is conditionally bisimilar to U_0 under the condition $\mathcal{A}_C$.

▶ **Example 4.10** (Unreliable channel vs. no channel)**.** For Examples 4.3 and 4.9, it can also be shown that under the condition $\neg\mathcal{A}_C$, the unreliable channel $\emptyset \to U_0 \leftarrow I_0$ is conditionally bisimilar to not having a channel between the two nodes $(\emptyset \to I_0 \leftarrow I_0)$.

In this case, U_0 can still do a reaction under $\mathcal{A}_C$. Then, I_0 can answer with an empty set of steps. The implication $\mathcal{A}_C \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i\in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$ is then simplified to $\mathcal{A}_C \wedge \neg\mathcal{A}_C \models \text{false}$, which is easily seen to be valid.

5 Up-to Techniques for Proving Conditional Bisimilarity

Our optimizations so far involved replacing context steps by representative steps, which ensure finite branching and thus greatly reduce the proof obligations for a single step. However, it can still happen very easily that the smallest possible bisimulation is of infinite size, in which case automated proving of conditional bisimilarity becomes impossible. For instance, in Example 4.9, the least conditional bisimulation relating the two cospans u, r (representing (un)reliable channels) contains infinitely many triples $(u;m^n,\ r;m^n,\ \mathcal{A}_C)$ for any number n of messages on the right node $(m = I_0 \to I_r \leftarrow I_0)$.

On the other hand, conditional bisimilarity is closed under contextualization, hence if u, r are related, we can conclude that $u;m$ and $r;m$ must be related as well. Intuitively the relation $R = \{(u, r, \mathcal{A}_C)\}$ is a sufficient witness, since after one step we reach the triple $(u;m,\ r;m,\ \mathcal{A}_C)$, from which we can "peel off" a common context m to obtain a triple already contained in R.

This is an instance of an *up-to technique*, which can be used to obtain smaller witness relations by identifying and removing redundant elements from a bisimulation relation. Instead of requiring the redundant triple $(u;m,\ r;m,\ \mathcal{A}_C)$ to be contained in the relation, it is sufficient to say that *up to* the passive context m, the triple is represented by $(u, r, \mathcal{A}_C)$, which is already contained in the relation. In particular, this specific up-to technique is known as *up-to context* [25], a well-known proof technique for process calculi.

Note that in general, a bisimulation up-to context is not a bisimulation relation. However, it can be converted into a bisimulation by closing it under all contexts.

In this section, we show how to adapt this concept to conditional bisimilarity and in particular discuss how to deal with the conditions in a conditional bisimulation up-to context.

5.1 Conditional Bisimilarity Up-To Context

We start our investigation of conditional bisimilarity up-to context with the idea of a relation that can be extended to a conditional bisimulation. To show, using such a conditional bisimulation up-to context R, that a pair of arrows is conditionally bisimilar, one cannot necessarily find the pair in R, but instead extends a pair in R to the pair under review. As this extension might provide parts of the context that the original condition referred to, it is necessary to shift the associated condition over the extension.

▶ **Definition 5.1** (Conditional bisimulation up-to context (CBUC)). *A symmetric conditional relation R is a* conditional bisimulation up-to context *if the following holds: for each triple $(a, b, \mathcal{C}) \in R$ and each context step $a \xrightarrow{f,\mathcal{A}}_C a'$, there are answering steps $b \xrightarrow{f,\mathcal{B}_i}_C b'_i$, $i \in I$, and conditions $\mathcal{C}''_i$ such that for each $i \in I$ there exists $(a''_i, b''_i, \mathcal{C}''_i) \in R$ with $a' = a''_i;j_i$, $b'_i = b''_i;j_i$ for some arrow j_i and additionally $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i\in I} \left(\mathcal{C}''_{i\downarrow j_i} \wedge \mathcal{B}_i\right)$.*

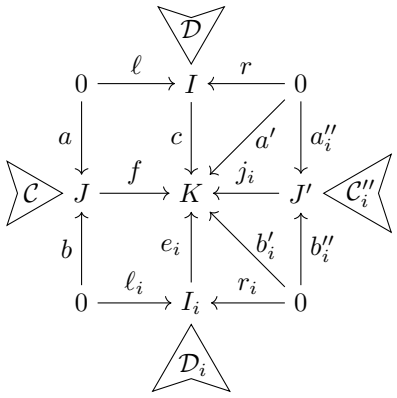


Figure 5 A single answer step in conditional bisimulation up-to context.

The situation for one answer step is depicted in Figure 5. The conditions $\mathcal{A}, \mathcal{B}_i$ over K are not shown in the diagram. The weakest possible $\mathcal{A}, \mathcal{B}_i$ can be derived from the rule conditions as $\mathcal{A} = \mathcal{D}_{\downarrow c},\ B_i = \mathcal{D}_{i\downarrow e_i}$.

Compared to a regular conditional bisimulation, which directly relates the results of the answering steps $(a', b'_i, \mathcal{C}'_i)$, in a CBUC it is sufficient to relate some pair $(a''_i, b''_i, \mathcal{C}''_i)$, where a''_i, b''_i are obtained from a', b'_i by removing an identical context j_i.

We now show that this up-to technique is useful or *sound*, that is, all elements recognized as bisimilar by the up-to technique are actually bisimilar [26, 25].

▶ **Theorem 5.2** (Characterization of CBUC). *A symmetric conditional relation R satisfies Definition 5.1 (is a CBUC) iff its closure under contextualization $\hat{R} := \{(a;d, b;d, \mathcal{C}_{\downarrow d}) \mid (a, b, \mathcal{C}) \in R,\ a, b\colon 0 \to J,\ d\colon J \to K\}$ is a conditional bisimulation.*

▶ Remark 5.3. From Theorem 5.2 we easily obtain as a corollary that every CBUC R is contained in $\overset{\circ}{\sim}_C$ ($R \subseteq \overset{\circ}{\sim}_C$), i.e. all elements contained in some CBUC are indeed conditionally bisimilar. This follows from the fact that $R \subseteq \hat{R}$ (set $d = \mathrm{id}_J$) and $\hat{R} \subseteq \overset{\circ}{\sim}_C$ (since by Theorem 5.2 $\hat{R}$ is a conditional bisimulation).

Note that while Theorem 5.2 gives a more accessible definition of CBUCs than Definition 5.1, the latter definition is amenable to mechanization, since R might be finite, whereas $\hat{R}$ is infinite.

5.2 Conditional Bisimilarity Up-To Context with Representative Steps

CBUCs allow us to represent certain infinite bisimulation relations in a finite way. For instance, we can use a finite CBUC in Example 4.9. However, automated checking for conditional bisimilarity up-to context is still hard, since all possible context steps have to be checked, of which there can be infinitely many.

For conditional bisimulations, we introduced an alternative definition using representative steps (Definition 4.7) and showed that it yields an equivalent notion of conditional bisimilarity (Theorem 4.8). We will show that the same approach can be used for CBUCs.

▶ **Definition 5.4** (CBUC with representative steps). *A* CBUC with representative steps *is a symmetric conditional relation R such that the following holds: for each triple $(a, b, \mathcal{C}) \in R$ and each representative step $a \xrightarrow{f,\mathcal{A}}_R a'$, there are answering steps $b \xrightarrow{f,\mathcal{B}_i}_C b'_i$ and conditions $\mathcal{C}''_i$ such that for each answering step there exists $(a''_i, b''_i, \mathcal{C}''_i) \in R$ with $a' = a''_i;j_i$, $b'_i = b''_i;j_i$ for some arrow j_i per answering step, and additionally $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i\in I} (\mathcal{C}''_{i\downarrow j_i} \wedge \mathcal{B}_i)$.*

▶ **Theorem 5.5.** *A conditional relation is a CBUC (Definition 5.1) if and only if it is a CBUC with representative steps (Definition 5.4).*

▶ **Example 5.6.** Consider again Examples 4.3 and 4.9. We have previously seen that it is possible to repeatedly borrow a message on the left-hand node and transfer it to the right-hand node, which leads to more and more received messages accumulating at the right-hand node. We now show that the two types of channels are conditionally bisimilar by showing that $R = \{(\emptyset \to R_0 \leftarrow I_0,\ \emptyset \to U_0 \leftarrow I_0,\ \mathcal{A}_C)\}$ is a CBUC, i.e. it satisfies Definition 5.4. We consider the same steps as in Example 4.9:

- R_0 can do a step using rule P_R by borrowing a message on the left node, with environment condition $\mathcal{A} = \text{true}$, and reduces to $a' = \emptyset \to R_r \leftarrow I_0$. U_0 can answer this step using P_U under $\mathcal{B}_i = \mathcal{A}_C$ (no noise) and reacts to $b'_i = \emptyset \to U_r \leftarrow I_0$.
  Now set $j_i = I_0 \to I_r \leftarrow I_0$, i.e. we consider the m-loop on the right node as irrelevant context. Then, using $a''_i = \emptyset \to R_0 \leftarrow I_0$, $b''_i = \emptyset \to U_0 \leftarrow I_0$, $\mathcal{C}''_i = \mathcal{A}_C$ we have $a' = a''_i;j_i$, $b'_i = b''_i;j_i$, and we find that the triple without the irrelevant context j_i, that is $(a''_i, b''_i, \mathcal{C}''_i)$ (which happens to be the same as our initial triple), is contained in R. As before, the implication $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i\in I} (\mathcal{C}''_i \wedge \mathcal{B}_i)$ holds.
- Symmetrically, U_0 borrows a message on the left node and reacts to U_r under $\mathcal{A} = \mathcal{A}_C$. Analogously to the previous case and to Example 4.9, R_0 answers this step, using $\mathcal{C}''_i = \mathcal{A}_C$ and $j_i = I_0 \to I_r \leftarrow I_0$.
- Again, the remaining representative steps can be proven in an analogous way.

Note that instead of working with an infinite bisimulation, we now have a singleton.

6 Comparison and An Alternative Characterization

6.1 An Equivalent Characterization Based on Environment Steps

We will now give a more natural characterization of conditional bisimilarity, in order to justify Definitions 4.2 and 4.7. This alternative definition is more elegant since it characterizes $\overset{\circ}{\sim}_C$ as the largest conditional congruence that is a conditional environment bisimulation. On the other hand, this definition is not directly suitable for mechanization.

In [16], environment steps, which capture the idea that a reaction is possible under some *passive* context d, have been defined to obtain a more natural characterization of saturated bisimilarity. Unlike the borrowed context f, the passive context d does not participate in the reaction itself, but we refer to it to ensure that the application condition of the rule holds.

▶ **Definition 6.1** (Environment step [16]). *Let $\mathcal{R}$ be a set of reactive system rules and $a\colon 0 \to K,\ a'\colon 0 \to K,\ d\colon K \to J$ be arrows. We write $a \overset{d}{\rightsquigarrow} a'$ whenever there exists a rule $(\ell, r, \mathcal{B}) \in \mathcal{R}$ and an arrow c such that $a = \ell;c,\ a' = r;c$ and $c;d \models \mathcal{B}$.*

Environment steps and context steps are related: they can be transformed into each other. Furthermore saturated bisimilarity is the coarsest bisimulation relation over environment steps that is also a congruence [16]. We now give a characterization of conditional bisimilarity based on environment steps:

▶ **Definition 6.2** (Conditional environment congruence). *A symmetric conditional relation R is a* conditional environment bisimulation *if whenever $(a, b, \mathcal{C}) \in R$ and $a \overset{d}{\rightsquigarrow} a'$ for some $d \models \mathcal{C}$, then $b \overset{d}{\rightsquigarrow} b'$ and $(a', b', \mathcal{C}') \in R$ for some condition $\mathcal{C}'$ such that $d \models \mathcal{C}'$. We denote by $\overset{\circ}{\sim}_E$ the largest conditional environment bisimulation that is also a conditional congruence and call it* conditional environment congruence.

▶ **Theorem 6.3.** *Conditional bisimilarity and conditional environment congruence coincide, that is, $\overset{\circ}{\sim}_C = \overset{\circ}{\sim}_E$.*

6.2 Comparison to Other Equivalences

We conclude this section by considering $\overset{\circ}{\sim}_T := \{(a, b) \mid (a, b, \text{true}) \in \overset{\circ}{\sim}_C\}$, a binary relation derived from conditional bisimilarity, which is ternary. Intuitively it contains pairs (a, b), where a, b are system states that behave equivalently in every possible context. We investigate how $\overset{\circ}{\sim}_T$ compares to other behavioural equivalences that also check for identical behaviour in all contexts. First, we consider saturated bisimilarity ($\sim_C$), which has been characterized in [16] as the coarsest relation which is a congruence as well as a bisimilarity:

▶ **Theorem 6.4.** *Saturated bisimilarity implies* true-*conditional bisimilarity ($\sim_C \subseteq \overset{\circ}{\sim}_T$). However,* true-*conditional bisimilarity does* not *imply saturated bisimilarity ($\overset{\circ}{\sim}_T \not\subseteq \sim_C$).*

For saturated bisimilarity, if a step of a is answered by b with multiple steps, all b'_i reached in this way must be saturated bisimilar to a' (that is, show the same behaviour even if the environment is later changed to one which did not allow the given b'_i to be reached). In fact, it was an explicit goal in the design of saturated bisimilarity to account for external modification of the environment.

On the other hand, for conditional bisimilarity, each b'_i is only required to be conditionally bisimilar to a' under the condition which allowed this particular answering step – that is, after a step, the environment is fixed (or, depending on the system, can only assume a subset of all possible environments, cf. Definition 6.2 and Theorem 6.3).

Next, we compare $\overset{\circ}{\sim}_T$ to id-congruence, the coarsest congruence contained in bisimilarity over the reaction relation $\rightsquigarrow$. It simply relates two agents whenever they are bisimilar in all contexts, i.e. $\sim_{\text{id}} := \{(a, b) \mid \text{for all contexts } d,\ a;d,\ b;d \text{ are bisimilar wrt. } \rightsquigarrow\}$.

▶ **Theorem 6.5.** *It holds that* true-*conditional bisimilarity implies* id-*congruence ($\overset{\circ}{\sim}_T \subseteq \sim_{\text{id}}$). However,* id-*congruence does* not *imply* true-*conditional bisimilarity ($\sim_{\text{id}} \not\subseteq \overset{\circ}{\sim}_T$).*

Intuitively, true-conditional bisimilarity allows to observe whether some item is consumed and recreated (by including it in both sides of a rule) or whether it is simply required (using an existential rule condition, cf. Theorem 6.5). On the other hand, id-congruence does not recognize this and simply checks whether reactions are possible in the same set of contexts.

Hence we have $\sim_C \subsetneq \overset{\circ}{\sim}_T \subsetneq \sim_{\mathrm{id}}$, which implies that checking for identical behaviour in all contexts using conditional bisimilarity gives rise to a new kind of behavioural equivalence, which does not allow arbitrary changes to the environment (as $\sim_C$ does), yet allows distinguishing borrowed and passive context (which $\sim_{\mathrm{id}}$ does not). For two of those equivalences ($\sim_C$, $\overset{\circ}{\sim}_T$) we can mechanize bisimulation proofs.

7 Conclusion, Related and Future Work

As stated earlier, there are some scattered approaches to notions of behavioural equivalence that can be compared to conditional bisimilarity. The concept of behaviour depending on a context is also present in Larsen's PhD thesis [21]. There, the idea is to embed an LTS into an environment, which is modelled as an action transducer, an LTS that consumes transitions of the system under investigation – similar to CCS synchronization. He then defines environment-parameterized bisimulation by considering only those transitions that are consumed in a certain environment. In [15], Hennessy and Lin describe symbolic bisimulations in the setting of value-passing processes, where Boolean expressions restrict the interpretations for which one shows bisimilarity. Instead in [2], Baldan, Bracciali and Bruni propose bisimilarity on open systems, specified by terms with a hole or place-holder. Instead of imposing conditions on the environment, they restrict the components that are filling the holes.

In [11], Fitting studies a matrix view of unlabelled transition system, annotated by Boolean conditions. In [3] we have shown that such systems can alternatively be viewed as conditional transition systems, where activation of transitions depends on conditions of the environment and one can state the bisimilarity of two states provided that the environment meets certain requirements. This view is closely tied to featured transition systems, which have been studied extensively in the software engineering literature. The idea here is to specify system behaviour dependent on the features that are present in the product (see for instance [7] for simulations on featured transition systems).

Our contribution in this paper is to consider conditional bisimilarity based on contextualization in a rule-based setting. That is, system behaviour is specified by generic rewriting rules, system states can be composed with a context specifying the environment and we impose restrictions on those contexts. By viewing both system states and contexts as arrows of a category, we can work in the framework of reactive systems à la Leifer and Milner and define a general theory of conditional bisimilarity. While in [16] conditions were only used to restrict applicability of the rules and bisimilarity was checked for all contexts, we here additionally use conditions to establish behavioural equivalence only in specific contexts.

As future work we want to take a closer look at the logic that we used to specify conditions. Conditional bisimilarity is defined in a way that is largely independent of the kind of logic, provided that the logic supports Boolean operators and shift. It is unclear and worth exploring whether the logic considered by us is expressive enough to characterize all contexts that ensure bisimilarity of two given arrows.

Up-to techniques can be elegantly stated in a lattice-theoretical framework [24] and it is not difficult to reframe the results of Section 5 in this setting, using the notion of compatibility. This view might help to incorporate further optimizations into the up-to technique.

Furthermore, it is an open question whether there is an alternative characterization of the id-congruence of Theorem 6.5 that is amenable to mechanization.

We have already implemented label derivation and bisimulation checking in the borrowed context approach, see for instance [23], however without taking conditions into account. Our aim is to obtain an efficient implementation for the scenario described in this paper. Note

that our conditions subsume first-order logic [6] and hence in order to come to terms with the undecidability of implication we have to resort to simpler conditions or use approximative methods.

One natural question is whether our results can be stated in a coalgebraic setting, since coalgebra provides a generic framework for behavioural equivalences. We have already studied a much simplified coalgebraic version of conditional systems (without considering contextualization) in [1], using coalgebras living in Kleisli categories. Reactive systems can also be viewed as coalgebras (see [4]). However, a combination of these features has not yet been considered as far as we know.

References

1 Jiří Adámek, Filippo Bonchi, Mathias Hülsbusch, Barbara König, Stefan Milius, and Alexandra Silva. A coalgebraic perspective on minimization and determinization. In *Proc. of FOSSACS '12*, pages 58–73. Springer, 2012. LNCS/ARCoSS 7213.

2 Paolo Baldan, Andrea Bracciali, and Roberto Bruni. Bisimulation by unification. In *Proc. of AMAST '02*, pages 254–270. Springer, 2002. LNCS 2422.

3 Harsh Beohar, Barbara König, Sebastian Küpper, and Alexandra Silva. Conditional transition systems with upgrades. In *Proc. of TASE '17 (Theoretical Aspects of Software Engineering)*. IEEE Xplore, 2017.

4 Filippo Bonchi. *Abstract Semantics by Observable Contexts*. PhD thesis, Università degli Studi di Pisa, Dipartimento di Informatica, May 2008.

5 Filippo Bonchi, Barbara König, and Ugo Montanari. Saturated semantics for reactive systems. In *Proc. of LICS '06*, pages 69–80. IEEE, 2006.

6 H.J. Sander Bruggink, Raphaël Cauderlier, Mathias Hülsbusch, and Barbara König. Conditional reactive systems. In *Proc. of FSTTCS '11*, volume 13 of *LIPIcs*. Schloss Dagstuhl – Leibniz Center for Informatics, 2011.

7 Maxime Cordy, Andreas Classen, Gilles Perrouin, Pierre-Yves Schobbens, Patrick Heymans, and Axel Legay. Simulation-based abstractions for software product-line model checking. In *Proc. of ICSE '12*, pages 672–682. IEEE, 2012.

8 Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, and Michael Löwe. Algebraic approaches to graph transformation—part I: Basic concepts and double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations*, chapter 3. World Scientific, 1997.

9 Hartmut Ehrig and Barbara König. Deriving bisimulation congruences in the DPO approach to graph rewriting. In *Proc. of FOSSACS '04*, pages 151–166. Springer, 2004. LNCS 2987.

10 Hartmut Ehrig, Michael Pfender, and Hans Jürgen Schneider. Graph-grammars: An algebraic approach. In *14th Annual Symposium on Switching and Automata Theory (SWAT 1973)*, pages 167–180, October 1973.

11 Melvin Fitting. Bisimulations and boolean vectors. In *Advances in Modal Logic*, volume 4, pages 1–29. World Scientific Publishing, 2002.

12 Annegret Habel, Reiko Heckel, and Gabriele Taentzer. Graph grammars with negative application conditions. *Fundamenta Informaticae*, 26(3,4):287–313, December 1996.

13 Annegret Habel, Jürgen Müller, and Detlef Plump. Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science*, 11(5):637–688, October 2001.

14 Annegret Habel and Karl-Heinz Pennemann. Correctness of high-level transformation systems relative to nested conditions. *Mathematical Structures in Computer Science*, 19(2):245–296, 2009.

15 Matthew Hennessy and Huimin Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.

16 Mathias Hülsbusch and Barbara König. Deriving bisimulation congruences for conditional reactive systems. In *Proc. of FOSSACS '12*, pages 361–375. Springer, 2012. LNCS/ARCoSS 7213.

17 Mathias Hülsbusch, Barbara König, Sebastian Küpper, and Lars Stoltenow. Conditional bisimilarity for reactive systems, 2020. arXiv:2004.11792. URL: `https://arxiv.org/abs/2004.11792`.

18 Ole Høgh Jensen and Robin Milner. Bigraphs and transitions. In *Proc. of POPL 2003*, pages 38–49. ACM, 2003.

19 Bartek Klin, Vladimiro Sassone, and Paweł Sobociński. Labels from reductions: towards a general theory. In *Proc. of CALCO '05*, pages 30–50. Springer, 2005. LNCS 3629.

20 Stephen Lack and Paweł Sobociński. Adhesive and quasiadhesive categories. *RAIRO – Theoretical Informatics and Applications*, 39(3):511–545, 2005.

21 Kim Guldstrand Larsen. *Context-Dependent Bisimulation between Processes*. PhD thesis, University of Edinburgh, 1986.

22 James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In *CONCUR 2000 — Concurrency Theory: 11th International Conference University Park, PA, USA, August 22–25, 2000 Proceedings*, pages 243–258. Springer Berlin Heidelberg, 2000.

23 Dennis Nolte. Automatischer Nachweis von Bisimulationsäquivalenzen bei Graphtransformationssystemen. Master's thesis, Universität Duisburg-Essen, November 2012.

24 Damien Pous. Complete lattices and up-to techniques. In *Proc. of APLAS '07*, pages 351–366. Springer, 2007. LNCS 4807.

25 Damien Pous and Davide Sangiorgi. Enhancements of the coinductive proof method. In Davide Sangiorgi and Jan Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*. Cambridge University Press, 2011.

26 Davide Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(5):447–479, 1998.

27 Vladimiro Sassone and Paweł Sobociński. Reactive systems over cospans. In *Proc. of LICS '05*, pages 311–320. IEEE, 2005.

28 Paweł Sobociński. *Deriving process congruences from reaction rules*. PhD thesis, University of Aarhus, 2004.